

Using System Dynamics Simulation Models for Software Project Management Education and Training

Dietmar Pfahl¹, Marco Klemm², Günther Ruhe¹

¹Fraunhofer Institute for Experimental Software Engineering (IESE)

²University of Kaiserslautern

Extended Abstract

Introduction

Software development is a dynamic and complex process as there are many interacting factors throughout the lifecycle that impact cost and schedule of the development project and quality of the developed software product. In addition, software industry constantly faces increasing demands for quality, productivity, and time-to-market, thus making the management of software development projects one of the most difficult and challenging tasks in any software organisation. Therefore, it is not surprising, that project management is one of the focus areas to which process simulation techniques have been applied in the domain of software engineering during the last decade, starting with the pioneering work of Kellner et al. [KeH89][HuK89] and Abdel-Hamid and Madnick [AbM91].

Considering that the need for software is constantly growing world-wide, and hence the need for experienced and well-trained project managers, it is surprising that experience with using process simulation as a means for software project management education and training has rarely been published (recent examples are [DrL99] and [MaT99]).

The objective of this paper is to present concepts of a computer based training (CBT) module for student education in software project management. The CBT module can be run using standard web-browsers (e.g. Netscape). The simulation component of the CBT module is implemented using the System Dynamics (SD) simulation modelling method. The paper will present the design of the simulation model and the training scenario offered by the existing single-learner prototype CBT module. Possibilities for empirical validation of the effectiveness of the CBT in university education will be discussed, and future extensions of the CBT module towards collaborative learning environments will be suggested.

Motivation and background

There is an increasing demand for software project managers in industry. Therefore, efforts are needed to develop the management-related knowledge and skills of the current and future software workforce. In particular, university education needs to provide to their computer science and software engineering (SE) students not only technology-related skills but in addition a basic understanding of typical phenomena occurring in industrial (and even academic) software projects.

The potential of simulation models for the training of managers has long been recognised: flight-simulator-type environments (or microworlds) confront managers with realistic situations that they may encounter in practice, and allow them to develop experience without the risks incurred in the real world. Two detailed examples of training workshops based on real industrial cases using simulation can be found in [Gra92] (other examples are mentioned in [Mor88] and [Mil98], to give a few pointers).

As regards the specific topic of *software* project management, experimental studies have been conducted on using simulation models representing the typical behaviour of software development projects.

Experiments carried out at the Jet Propulsion Laboratory aimed at studying the decision-making process of software managers [Lin93]. Twenty managers were asked to conduct a project simulated with the aid of the Software-Engineering Process Simulation Model (SEPS) model [LAS97]; some were provided with cause-effect feedback of their actions, while the others were not. It was observed that the second group (without feedback information) tended

to act in a more "fire fighting" mode than the first one; and the feedback information was most beneficial to the less experienced managers.

At Draper Laboratory, a simulation model served as the basis for an experiment involving a group of 50 experienced managers [SNV93]. The scenario of the experiment involved a 15 percent requirement change in the course of the simulated project. Few of the managers were able to adapt properly to this situation: most of them reacted by hiring new staff late on the project (a typical "fire fighting" policy), and experienced budget and schedule overruns. Worse, the managers tended to reproduce exactly the same errors when running the scenario for the fourth or fifth time. The authors of the study on the experiment conclude that the participating managers were limited by their mental model of the process, *and* that they were reluctant to change it.

The two experiments reported above show that natural one-way causal thinking can be detrimental to the success of software managers. Therefore, the aim of the training should go beyond that of facing people to realistic problems; the concern is also to make managers adopt systems thinking, and perceive the existence of (unexpected) feedback to management decisions. This sets the problem of an adequate game interface and more importantly of the workshop or training course organisation: just running a simulation model as a black box may not be sufficient to have people gain insight into the software development process, and accept to alter their mental model. Lessons learned from experiments in strategic management training [Gra92] indicate that adequate course organisation avoids the situation observed at Draper Laboratory, where managers played the simulation game without analysing their errors, hence gaining insufficient insight into the problem.

Objectives

The main objective of the research reported in this paper is to build a simulation-based CBT module that helps SE students (or – after enhancements – candidates for project management positions in industry) understand the complex decision-making situations in project management, which are characterised by trade-off effects between (usually conflicting) goals. Typical project management goals are to shorten the project duration (time-to-market), to decrease the project cost, and/or to improve the quality of the delivered end product (for specified product functionality). Using the characterisation grid published by Kellner et al. [KMR99], the process simulation model contained in the prototype CBT module, which consists of a training scenario *and* the integrated simulation model, can be classified as shown in Table 1.

Table 1. Characterisation grid to classify process simulation models with regards to purpose and scope.

Purpose \ Scope	Scope	Portion of life-cycle	Development project	Multiple, concurrent projects	Long-term product evolution	Long-term organisation
Strategic management						
Planning						
Control and operational management						
Process improvement and technology adoption						
Understanding						
Training and learning			duration, effort, size, field defect density			

The purpose of the CBT module is to support training of SE students. The courseware will mainly focus on managerial aspects related to the performance of a complete software development project (dark grey area). There are, however, possibilities to extend the simulation model and the associated training scenario such that individual development phases or multiple project interrelations are stressed (light grey area). The key result variables of the simulation model represent project duration, effort consumption, product size, and product quality after system test.

CBT module contents

The goal of the CBT module is to transfer basic knowledge about software project management to computer science and SE students by providing a scenario-driven interactive single-learner environment that can be activated through a standard web-browser. The core element of the CBT module is a plug-in that provides the functionality of an SD model that simulates the typical behaviour of software development projects. The features of the simulation model

(CBT/Simulator) and the design of the single-learner training scenario (CBT/Scenario) will be fully described in the paper.

Features of the CBT/Simulator (outline)

The structure of the simulation model represents in a simplified, generic waterfall-model-like fashion three phases of a typical software development project: Design, Implementation, and Test. The calibration of the model was not based on a real industrial case or on exhaustive empirical research, but on the functional relationships between effort, time, and size, as suggested by the well-known COCOMO model [Boe81].

In total, the simulation model consists of five interrelated sub-models (views):

- **Production:** This view represents a typical software development lifecycle consisting of the following chain of transitions: set of requirements → design documents → code → tested code. Note that the detection of defects during testing only causes reworking of the code.
- **Quality:** In this view, the defect co-flow is modelled, i.e.: defect injection (into design or code) → defect propagation (from design to code) → defect detection (in the code during testing)→ defect correction (only in the code).
- **Effort:** In this view, the total effort consumption for design development, code development, code testing, and defect correction (i.e. rework) is calculated.
- **Initial Calculations:** In this view, using the COCOMO equations, the normal value of the central process parameter “productivity” is calculated. The normal productivity varies with assumptions about the product development mode (organic, semi-detached, embedded) and characteristics of the project resources available (e.g. developer skill).
- **Productivity, Quality & Manpower Adjustment:** In this view, project-specific process parameters, like (actual) productivity, defect generation, effectiveness of QA activities, etc., are determined based on a) planned target values for manpower, project duration, product quality, etc., and b) time pressure induced by unexpected rework or changes in the set of requirements.

The most important model parameters are listed in Table 2. It should be noted, however, that the model user could choose from more than 30 parameters, if required by more complex CBT scenarios.

Table 2: CBT/Simulator model parameters.

Input Parameters			Output Parameters
Project Characterisation Parameters (mandatory)	Project Management Parameters (optional)	QA technology-related Parameters (optional)	
Initial_job_size_in_tasks			Job_size_in_tasks
Project_complexity	Planned_completion_time		Project_completion_time
	Planned_manpower		Effort
	Manpower_skill		
	Goal_field_defect_density	Application of design and/or code inspections	Field_defect_density

For the implementation of the simulation model the SD modelling tool Vensim 3.0 was used [Ven97], thus providing a set of instructive graphical analysis functions to the model user. The user interface was developed in ObjectPascal.

Design of the CBT/Scenario (outline)

The training scenario has been designed as a single-learner interactive web-based courseware. The goal of the scenario is to make the trainee understand the complex implications of a set of principles that dominate software projects conducted according to the waterfall process model. Even though the waterfall process approach is no longer state-of-the-art and in most industrial organisations not even state-of-the-practice, it is still an interesting object of study for students having little or no experience with real-world industrial software development. The set of principles (cf. Table 3) was distilled from the top 10 list of software metric relationships published by Boehm [Boe87].

In order to make the trainee understand the implications of these principles (and their combinations), in the scenario he has to take the role of a project manager who has been assigned to a new development project. Several constraints

are set, i.e. the size of the product and its quality requirements, the number of software developers available, and last but not least the project deadline. The first thing to do for the project manager (i.e. in order to familiarise with the simulation model) is to check whether the project deadline is feasible under the resource and quality constraints given. Soon, the project manager learns that the deadline is much too short. Now, the scenario provides a set of actions that the project manager can take, each action associated with one of the principles and linked to one of the model parameters listed in Table 2. Soon the project manager will learn that his department head does not accept all of the proposed actions (e.g. reducing the product size or complexity). Depending on what action the project manager has chosen additional options can be taken. Eventually, the project manager will find a way to meet the planned deadline, e.g. by introducing code and design inspections. In any case, he will only succeed when combining actions that relate to at least two of the principles listed in table 3.

Table 3. List of principles dominating project performance.

No.	Principle
1	"Finding and fixing a software problem after delivery is 100 times more expensive than finding and fixing it during the requirements and early design phases."
2	"You can compress a software development schedule up to 25 percent of nominal, but no more."
3	"Software development and maintenance cost are primarily a function of the number of source lines of code (SLOC) in the product."
4	"Variations between people account for the biggest differences in software productivity."
5	"Software systems and products typically cost 3 times as much per SLOC as individual software programs. Software-system products (i.e. system of systems) cost 9 times as much."
6	"Walkthroughs catch 60% of the errors."

In order to support the learn effect of the scenario, and to avoid the phenomena observed at Draper Laboratory, the trainee will be properly briefed before and de-briefed after running the CBT [Lan95].

Validation

The validation of the CBT module consists of two steps. The first step has already been concluded, the second step is in the planning stage.

Step 1 aims at validating the accuracy and usability of the simulation model. The accuracy of the simulation model has been proven through comparison of the simulation results with estimates of the COCOMO model (organic mode). The usability has been proven through successful application in several presentations and lectures on SW project management.

Step 2 aims at validating the effectiveness of the CBT module. Submitting the CBT module to evaluation in a controlled experiment with students will test this. In the experiment, the project management related knowledge gained by students that were trained with the CBT module is compared to the knowledge gained by another group of students that were trained by simply reading a text describing a similar case study in the same period of time. The design of the experiment will be outlined in the paper, and results of the experiment will be presented at the workshop.

Conclusion and Future Research

Based on the results of the second validation step, further enhancements of the CBT module and additional experiments will be conducted in order to improve the effectiveness

A main direction of research will pursue possibilities to extend the SD model (and the associated training scenario) towards collaborative learning/training.

Perspective 1: Instead of having only one active role (one "player") participating in the simulation of the software development project, i.e. the project manager with responsibility for the phases design, coding, and testing, there are three active roles involved, i.e. one sub-project (or phase) manager responsible for each development phase. Since decisions made by one sub-project manager might interfere with the decisions made by the others, communication and co-operation among sub-project managers is necessary. In order to guide the discussion among the "players", a fourth "player" might be involved, i.e. the overall project manager, in this case having the responsibility of a tutor who interferes discussions whenever necessary.

Perspective 2: Several stand-alone CBT sessions are coupled and run in parallel. In this scenario, there is still only one project manager responsible for one complete project (all three phases), but there are several of them acting in parallel,

thus simulating a software department that has several concurrent development projects ongoing. The synchronisation of the concurrent projects has to be done via a limited resource, i.e. the pool of available software developers in the department. In this scenario there could be added the role "department head" as an active "player" who is in charge of synchronising the development projects.

Both perspectives require modifications/extensions of the current CBT module. In addition, there is a need for integrating the CBT module with a distributed platform allowing for web-based communication and collaboration among several "players". Several such platforms have already been developed (e.g., HyperWave, MTS). Their use for collaborative learning/training will be investigated and enhanced in-depth in the recently started ESPRIT project CORONET [COR99].

References

- [AbM91] Abdel-Hamid TK, Madnick SE: Software Project Dynamics – an Integrated Approach, Prentice-Hall, 1991.
- [Boe81] Boehm BW, Software Engineering Economics, Prentice-Hall, 1981.
- [Boe87] Boehm BW: "Industrial software metrics top 10 list", IEEE Software, pp. 84-85, September 1987.
- [COR99] Corporate Software Engineering Knowledge Networks for Improved Training of the Work Force, 5th Framework ESPRIT Project of the European Commission, IST-1999-11634, 1999.
- [DrL99] Drappa A, Ludewig J: "Quantitative modeling for the interactive simulation of software projects", Journal of Systems and Software 46, pp. 113-122, 1999.
- [Gra92] Graham AK, Morecroft JDW, Senge PM, Sterman JD: "Model-supported case studies for management education", European Journal of Operational Research 59, pp. 151-166, 1992.
- [HuK89] Humphrey WS, Kellner MI: "Software Process Modeling: Principles of Entity Process Models", Proc. 11th Int'l Conf. Software Engineering (ICSE), IEEE Computer Soc., pp. 331-342, 1989.
- [KeH89] Kellner MI, Hansen GA: "Software Process Modeling: A Case Study", Proc. 22nd Annual Hawaii Int'l Conf. System Sciences, Jan. 1989.
- [KMR99] Kellner MI, Madachy RJ, Raffo DM: "Software process simulation modeling: Why? What? How?", Journal of Systems and Software 46, pp. 91-105, 1999.
- [Lan95] Lane DC: "On a Resurgence of Management Simulation Games", Journal of the Operational Research Society 46, pp. 604-625, 1995.
- [Lin93] Lin CY: "Walking on Battlefields: Tools for Strategic Software Management", American Programmer, pp. 33-40, May 1993.
- [LAS97] Lin CY, Abdel-Hamid T, Sherif JS: "Software-Engineering Process Simulation Model (SEPS)", Journal of Systems and Software 38, pp. 263-277, 1997.
- [MaT99] Madachy R, Tarbet D: "Case Studies in Software Process Modeling with System Dynamics", Proc. 2nd Software Process Simulation Modeling Workshop (ProSim'99), Silver Falls, Oregon, June 1999.
- [Mil95] Milling P: "Managementsimulation im Prozeß des Organisationalen Lernens" [Organisational Learning and its Support by Management Simulators], Zeitschrift für Betriebswirtschaft Ergänzungsheft 3/95: Lernende Unternehmen, pp. 93-112, March 1995. (Also available at URL <http://iswww.bwl.uni-mannheim.de>)
- [Mor88] Morecroft JDW: "System dynamics and microworlds for policymakers", European Journal of Operational Research 35, pp. 301-320, 1988.
- [SNV93] Smith BJ, Nguyen N, Vidale RF: "Death of a Software Manager: How to Avoid Career Suicide through Dynamic Software Process Modeling", American Programmer, pp. 10-17, May 1993.
- [Ven97] Ventana Simulation Environment (Vensim) - Reference Manual, Version 3.0A, Ventana Systems, Inc., 1997.